Universidad de las Ciencias Informáticas

Facultad 6

Estructura de datos espacial jerárquica para la indexación de agrupaciones de objetos geométricos

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Jairo Rojas Delgado
Tutor: MSc. Eddy Dangel Quesada Rodríguez

La Habana, 2015
Año 57 de la Revolución

Pensamiento

El que no aplique nuevos remedios, debe esperar nuevos males; porque el mayor innovador es el tiempo.

Francis Bacon (1561-1626)

Declaración de autoría

Declaro por este medio que yo, Jairo Rojas Delgado con carné de identidad 91060735307, soy el autor de este trabajo y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo. Para que así conste, firma la presente declaración jurada de autoría en La Habana a los ____ días del mes ____ del año _____.

Jairo Rojas Delgado	*Msc. Eddy Dangel Quesada Rodríguez*
Autor	**Tutor**

Datos de contacto

Tutor: MSc. Eddy Dangel Quesada Rodríguez.

Eddy Dangel Quesada Rodríguez, graduado en el año 2008 de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI), La Habana, Cuba. Se ha desempeñado como profesor de varias asignaturas docentes y optativas y tiene la categoría docente de profesor Asistente. El profesor también posee la categoría científica de Máster en Informática Aplicada y actualmente ocupa el cargo de Jefe de Departamento en el centro productivo GEYSED. El mismo cuenta con más de 5 años de experiencia en el desarrollo software asociado a las ramas de la Geología y la Minería, siendo autor de varias investigaciones publicadas en revistas y eventos científicos nacionales e internacionales.

Correo electrónico: edquezada@uci.cu

Dedicatoria

A los que modestamente construyen un mundo donde el talento humano sin excepciones encuentre manifestación. Al futuro de mi Patria.

Agradecimientos

Deseo agradecer a mi familia, profesores, amigos y compañeros que me han acompañado hasta este momento y han contribuido a mi formación profesional y humana. Familia, profesores y amigos son clasificaciones que se solapan, o esa al menos es mi experiencia. Así es como mis primeros mentores y amigos fueron mis dos abuelos.

Deseo agradecer en espacial:

A mi abuela y abuelo por el ejemplo y apoyo incondicional.

A mi mamá y papá. A mis tías Diana y Varina. A mis primos Norge y Dianelis. A mi hermana.

A mi tutor por su guía y enseñanzas.

Resumen

En la actualidad existen varios métodos para la obtención de información espacial digital del mundo real que comúnmente es expresada en términos de agrupaciones de objetos geométricos. Los sistemas mineros actuales deben procesar un gran volumen de datos, así como realizar un grupo operaciones complejas desde el punto de vista computacional, como es el caso de las búsquedas espaciales. Estas dos peculiaridades influyen de manera negativa en el comportamiento del tiempo de ejecución de estos sistemas y la investigación en esta área ha sido intensa en los últimos años. En este trabajo se desarrolla una estructura de datos espacial jerárquica que permita indexar objetos geométricos de forma balanceada, para disminuir la complejidad temporal de las búsquedas espaciales y de esta forma disminuir el tiempo de ejecución de las búsquedas espaciales realizadas en agrupaciones de objetos geométricos. Los resultados obtenidos muestran una disminución del tiempo de ejecución de las búsquedas espaciales en comparación con el empleo de una estructura de datos lineal, no obstante existe un incremento significativo del consumo de memoria RAM.

Palabras clave: búsqueda espacial, estructura de datos espacial jerárquica, tiempo de ejecución.

Abstract

Currently there are several methods for obtaining digital spatial data from the real world that is commonly expressed in terms of groups of geometric objects. Current mining systems must process a large volume of data and perform complex operations from the computational point of view, such as spatial searches. These two characteristics have a negative influence on the execution time behavior of these systems and research in this area has been intense in recent years. This paper presents a hierarchical spatial data structure that allows indexing geometric objects in a balanced way to reduce the time complexity of the spatial searches and thus decrease the execution time of spatial searches performed in geometric objects groups. The results obtained show a decrease of execution time of spatial searches compared to the use of a linear data structure, however there is a significant increase in consumption of RAM.

Key words: *spatial search, hierarchical spatial data structure, execution time.*

Índice general

Índice de figuras

Índice de tablas

Introducción

En los albores de la humanidad la extracción y uso de los minerales de la tierra constituyó un elemento imprescindible para la supervivencia del hombre, de hecho las minas más antiguas encontradas datan de más de 40 mil años según los registros oficiales. En el mundo contemporáneo se emplean cuantiosos recursos en la exploración y explotación del subsuelo. Las operaciones de explotación de minas implican el desembolso de importantes montos de inversión, la aplicación de operaciones y equipos de alta complejidad y el desarrollo de labores en condiciones extremas de trabajo. En este escenario, el uso de las herramientas de software adquiere una gran importancia durante todo el proceso de prospección y exploración minera, siendo el resultado de la aparición de nuevas y más convenientes formas de modelar el planeta.

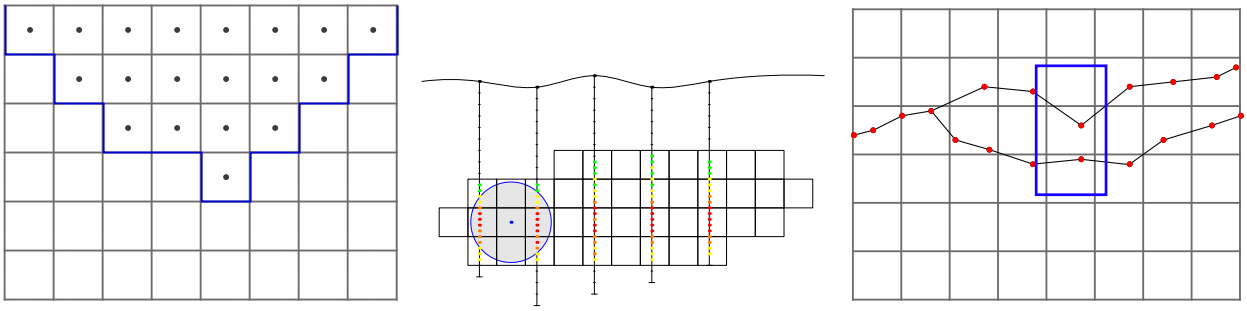
En Cuba se comienza a utilizar software con propósitos mineros a partir de 1983, donde sus aplicaciones fueron aisladas e incipientes y sin estar exentas de limitaciones de carácter material y financiero (Rodríguez, 2009). Durante la segunda mitad de la década de 1980 se extiende esta tecnología al Control Integral de Flujo de Datos para la industria del níquel (Ruiz, 2009). Con este precedente, a partir de 1995 se introduce un grupo de tecnologías informáticas de licencia privativa con altos costos de mantenimiento y capacitación, debido a que los procesos de informatización del país no lograron igualar el ritmo impuesto por el desarrollo tecnológico y las necesidades económicas y sociales (Legrá, 2003).

Motivado por los altos costos de la utilización de las tecnologías informáticas con licencia privativa y por políticas de soberanía tecnológica, en los últimos años ha existido la tendencia a migrar hacia soluciones de software minero construidas en el país. Debido a esto, en el proyecto Sistema Minero Cubano del centro de Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas se desarrolla el Sistema de Análisis y Modelado de Yacimientos Minerales (Syam), encargado de modelar tridimensionalmente la información asociada a yacimientos minerales.

Los sistemas mineros actuales, por lo general, deben realizar un grupo de operaciones complejas desde un punto de vista computacional. Entre estas operaciones puede mencionarse el problema de disminuir el costo computacional de procesar un modelo de bloques[1] geológicos para optimizar el diseño de una mina a cielo abierto (Quintana, 2013). Para construir dicho modelo de bloques se calcula por cada región espacial un valor mineral junto a un valor económico de cada bloque. Esto se hace con el objetivo de definir dentro del diseño de una mina abierta, el límite físico que tendrá la cantera, generalmente en forma de cono regular invertido de haciendo que se maximicen los beneficios a obtener, tal como puede apreciarse en la Figura(1a).

[1]Modelo de bloques: estructura mediante la cual se modelan computacionalmente los yacimientos minerales.

En las operaciones de estimación de recursos minerales, Figura(1b), se desea conocer el valor de la concentración mineral de una región espacial de la cual no se han obtenido mediciones explícitas. Para esto, se obtienen todas las mediciones realizadas en zonas cercanas a la región de interés y mediante un método de interpolación espacial es posible aproximar el valor de su concentración mineral. Otro ejemplo de operaciones realizadas en los sistemas mineros actuales se muestra en la Figura(1c) donde se desea conocer el volumen del material extraído contenido entre dos superficies de terreno.



(a) (b) (c)

Figura 1: *Operaciones comunes de los sistemas mineros actuales.*

Además de las operaciones espaciales mencionadas anteriormente, existe otro grupo de operaciones realizadas por estos sistemas enfocadas en la visualización tridimensional. En función de lograr un mayor aprovechamiento de las características de software y hardware se utilizan modelos de mallas y nubes de puntos. Esto a pesar de ser altamente soportado por el hardware y por tanto ser muy eficiente, la gran cantidad de datos a manejar y la complejidad de los algoritmos que se emplean usualmente rebasan los límites prácticos (Valle, 2009).

En Syam se realiza un grupo de complejos cálculos y operaciones de procesamiento de objetos geométricos como las búsquedas del vecino más cercano, las búsquedas regionales y otro grupo de operaciones espaciales. Estas operaciones por lo general implican realizar búsquedas espaciales involucrando cantidades de datos en el orden de los miles y millones de objetos geométricos y tienen la capacidad de consumir elevados tiempos de ejecución del sistema. A pesar de esto Syam no posee un mecanismo que permita agilizar las búsquedas espaciales contando con estructuras de datos lineales para procesarlas. Esta peculiaridad contribuye a que el consumo de tiempo de ejecución de las búsquedas espaciales se comporte de manera lineal con relación al volumen de datos procesado en la mayoría de los casos (Akenine-Möller et al., 2008).

Dada la situación problemática expuesta es posible definir el siguiente **problema de investigación**: ¿Cómo disminuir el tiempo de ejecución de las búsquedas espaciales realizadas en agrupaciones de objetos geométricos?

El problema permite definir como **objeto de estudio**: las búsquedas espaciales en agrupaciones de objetos geométricos y como **campo de acción**: las estructuras de datos jerárquicas para búsquedas espaciales en agrupaciones de objetos geométricos.

Para brindarle solución al problema se plantea como **objetivo general**: Desarrollar una estructura de

datos espacial jerárquica que permita indexar objetos geométricos de forma balanceada para disminuir la complejidad temporal de las búsquedas espaciales.

Después de haber realizado una revisión de la literatura y desarrollado consecuentemente el marco teórico se formula la siguiente **hipótesis de investigación**: Si se desarrolla una estructura de datos espacial jerárquica que permita indexar objetos geométricos de forma balanceada para disminuir la complejidad temporal de las búsquedas espaciales entonces será posible disminuir el tiempo de ejecución de las búsquedas espaciales realizadas en agrupaciones de objetos geométricos.

Para definir una estructura medible y cuantificable del progreso de la actividad investigativa se definen las siguientes **tareas de investigación**:

1. Caracterizar las diferentes técnicas de búsquedas espaciales en agrupaciones de objetos geométricos como mallas poligonales y nubes de puntos para obtener una medida del estado del arte relacionado con el objeto de estudio planteado.

2. Caracterizar las estructuras de datos espaciales para la indexación de mallas poligonales y nubes de puntos para obtener una medida del estado del arte relacionado con el campo de acción planteado.

3. Implementar una estructura de datos espacial para brindar una solución al problema de investigación en correspondencia con el objetivo planteado.

4. Caracterizar el comportamiento del consumo de memoria física de la estructura de datos implementada para definir los límites prácticos de su aplicación.

5. Caracterizar el comportamiento del consumo de tiempo de las búsquedas espaciales empleando la estructura de datos implementada y una estructura de datos lineal para demostrar la hipótesis de investigación.

Para dar cumplimiento al objetivo general y realizar las tareas de investigación, se han combinado un grupo de métodos y procedimientos teóricos y empíricos de la investigación científica, dentro de los cuales los fundamentales son:

Del nivel teórico

1. Analítico-Sintético. El método fue empleado para analizar el estado del arte de los procesos de búsqueda espacial y de esta forma obtener conocimiento procediendo a sintetizarlo.

2. Histórico-lógico. El método consiste en realizar una revisión exhaustiva del desarrollo evolutivo del objeto de investigación a lo largo del tiempo con el objetivo de definir las limitaciones actuales de su conocimiento. El método permitió reconocer los avances teórico-prácticos y problemas que actualmente existen en el área de investigación tratada.

3. Hipotético-deductivo. El método consiste en establecer una serie de verdades supuestas que deben ser comprobadas o rechazadas dando como resultado una hipótesis de investigación.

Del nivel empírico

1. Simulación. El método consiste en realizar una aproximación a la realidad de un fenómeno determinado. Esto significa que para emular dicho fenómeno es necesario un conjunto de datos que si bien no son reales se aproximan a los reales. Este método es utilizado para realizar las pruebas de la implementación de la estructura de datos resultante.

En función de orientar al lector, el presente trabajo se encuentra **estructurado** de la siguiente forma:

1. Introducción. Breve descripción del diseño metodológico de la investigación.
2. Fundamentos teóricos sobre las búsquedas espaciales. Ampliación del marco teórico sobre el proceso de búsqueda espacial realizado en mallas poligonales y en nubes de puntos.
3. Propuesta de solución para disminuir el tiempo de ejecución de las búsquedas espaciales. Se describen los procesos de indexación y búsqueda espacial empleando una estructura de datos *Octree*.
4. Pruebas y resultados de la estructura de datos *Octree*. Se muestra los resultados para el consumo de tiempo y memoria RAM al emplear una estructura de datos *Octree* y una estructura da datos lineal.
5. Conclusiones generales.
6. Referencias bibliográficas y bibliografía.

Capítulo 1

Fundamentos teóricos sobre las búsquedas espaciales

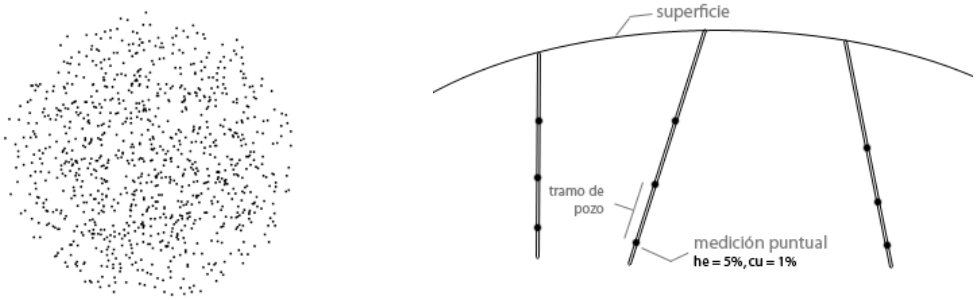
Planteado el problema de investigación, se procede a elaborar sus bases teóricas. Esto se hace con el objetivo de contrastar el panorama del conocimiento actual sobre el objeto de estudio expresado. Por tanto, en lo subsiguiente se exponen y analizan las teorías, conceptos y antecedentes relacionados con el proceso de modelado de información espacial en entornos digitales, las operaciones de búsqueda espacial y las estructuras de datos espaciales existentes.

1.1. Modelado de objetos tridimensionales

La construcción de objetos del mundo real en un entorno digital posee un marcado interés tanto en el ámbito científico como no científico (Zapotocky, 2013).Un método ampliamente utilizado para la construcción digital de objetos tridimensionales es el empleo de las primitivas geométricas. En esta investigación, una primitiva geométrica de forma general se refiere al objeto más simple, irreducible o atómico que un sistema puede manejar. La idea tras el uso de las primitivas geométricas, es utilizar su sencilla representatividad matemática para modelar objetos tridimensionales de mayor complejidad y de esta manera disminuir el tiempo de ejecución y consumo de memoria de cualquier computación posterior.

Existen diferentes técnicas para la obtención de información digital de objetos tridimensionales. El resultado final de muchas de estas técnicas es similar a una nube de puntos que representa la disposición de la superficie del objeto en cuestión tal como puede apreciarse en la Figura(1.1a). Este tipo de resultado es muy común en aplicaciones donde se realiza un escaneo mediante láser de un escenario tridimensional o cuando se asocia información geológica de tramos de pozos con puntos, tal como se aprecia en la Figura(1.1b). Por lo general es común que la cantidad de puntos generados que deben procesar los sistemas informáticos actuales oscile entre unos cuantos miles y millones.

Otra salida usual de estas técnicas para la obtención de información digital de objetos tridimensionales suele ser una malla poligonal. Muchos modelos de gráficos por computadoras se encuentran definidos usando mallas. Al unificar las nociones de malla poligonal de (Botsch et al., 2007, 2002; Joe, 1995), una malla



(a) (b)

Figura 1.1: *Nubes de puntos resultantes de aplicar diferentes técnicas para obtener información digital.*

poligonal es una superficie o un objeto poliédrico creados mediante un método tridimensional generado por sistemas de vértices, segmentos y caras posicionados en el espacio. La subdivisión de superficies es un método para representar una superficie suave mediante la especificación de una malla poligonal menos detallada. La superficie subdividida puede calcularse a partir de una malla más burda, iterando el proceso de dividir cada cara poligonal en caras más pequeñas que se aproximan mejor a la superficie suavizada[1].

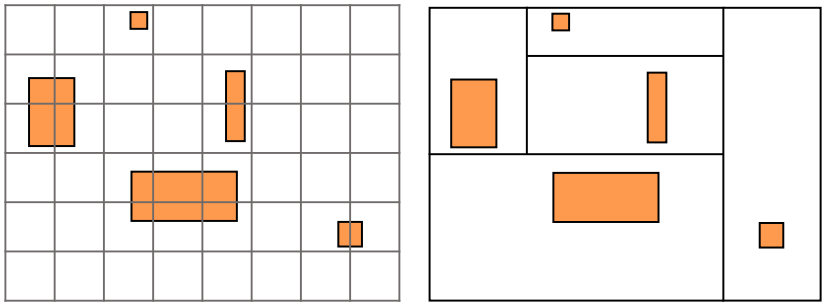
1.2. Proceso de particionado del espacio

En la presente investigación se entiende como agrupación de objetos geométricos, a un conjunto de primitivas dispuestas en el espacio, en lo subsiguiente un espacio euclidiano. Dichas agrupaciones de objetos geométricos, como las nubes de puntos o las mallas poligonales, generalmente involucran grandes cantidades de datos. Sin embargo, dichas agrupaciones de objetos geométricos pueden ser representadas listando pequeñas cantidades de puntos que le pertenecen (Hoffmann, 1989). Al proceso de listar las pequeñas porciones del espacio que pertenecen a una agrupación de objetos geométricos es lo que intuitivamente se conoce como proceso de particionado el espacio.

De manera formal, se define como proceso de particionado el espacio a la división del espacio en dos o más subconjuntos disjuntos (Mehta y Sahni, 2004). Según esto, una agrupación de objetos geométricos puede ser descompuesta en un conjunto discreto de elementos adyacentes que no se intersecan. Estos elementos son por lo general primitivas que pueden variar de forma, tamaño, posición y orientación. Para espacios tridimensionales, la más común de estas primitivas son los cubos y al utilizarlos se dice de un particionado regular del espacio puesto que son uniformes en forma, tamaño y orientación (Hoffmann, 1989). En la Figura (1.2a) se muestra un espacio bidimensional particionado de manera regular mediante cuadrados y en la Figura (1.2b) se presenta un particionado no regular del espacio puesto que las subregiones creadas no poseen el mismo tamaño ni forma.

Una vez representados estos objetos en un medio digital es de interés práctico realizar un grupo de

[1]Superficie suavizada: se refiere al modelo de superficie que más se aproxima al objeto real.



(a) (b)

Figura 1.2: *Particionado del espacio de forma regular y no regular.*

operaciones sobre esta información. Algunas de estas aunque no son de interés para la presente investigación son las operaciones booleanas[2] en modelado de sólidos (Requicha y Voelcker, 1985) y la simulación de fenómenos complejos que son difíciles de medir en el mundo real.

1.3. Técnicas de búsqueda espacial

En todo sistema de información es común realizar búsquedas que satisfagan las restricciones de una consulta a través de una gran cantidad de datos. La característica que distingue a las búsquedas espaciales es que dicha consulta se encuentra expresada en términos de localizaciones y relaciones espaciales. Las consultas espaciales pueden caracterizarse en dos grandes grupos: basadas en localización o basadas en fenómeno. Las consultas espaciales basadas en localización emplean la geometría inherente a los datos para devolver los objetos de interés. Las consultas espaciales basadas en fenómeno utilizan los atributos semánticos de los datos espaciales. Como un ejemplo de consulta espacial basada en fenómeno, considérese una consulta sobre una tupla de empleado con los atributos nombre, edad y sexo. Dicha tupla puede ser considerada como un punto tridimensional y sin embargo no posee ninguna relación espacial o geométrica (Mehta y Sahni, 2004). Siendo esto, las consultas espaciales basadas en localización son de mayor interés desde el punto de vista de la presente investigación.

A continuación se enuncian una serie de problemas comunes en el área de la geometría computacional en los que se emplea una búsqueda espacial como vía de solución. Tal es el caso de la búsqueda del vecino más cercano y de las búsquedas regionales.

1.3.1. Búsqueda del vecino más cercano

De acuerdo a (Chen, 1996), el problema de la búsqueda del vecino más cercano es familia de un grupo de problemas geométricos bien conocidos, tales como, el par más cercano, todos los vecinos cercanos, árboles

[2]Una operación booleana en el ámbito de modelado de sólidos hace referencia al proceso de obtener representaciones complejas de objetos tridimensionales mediante la resta, intersección y unión de solidos llamados primitivas.

de expansión mínimos, triangulación y círculo máximo vacío, los cuales suelen ser denominados como problemas de proximidad. Los problemas de proximidad surgen cuando un grupo de objetos tridimensionales son representados como puntos en el espacio como por ejemplo en algoritmos de *clustering*[3], clasificación o sistemas de control aéreo.

El problema de buscar el vecino más cercano puede ser definido formalmente como, dado un conjunto S de puntos en un espacio M y un punto de consulta $q \in M$ encontrar el punto más cercano a q en S (Knuth, 1998). La solución más sencilla a este problema es realizar una computación de la distancia euclidiana dada por la ecuación(1.1) del punto de consulta con el resto de los puntos que pertenecen a S manteniendo siempre la menor distancia encontrada. Este algoritmo, conocido como ingenuo posee una complejidad temporal de $O(n \times d)$ donde n es la cantidad de elementos de S y d la dimensión del espacio de búsqueda M (Weber et al., 1998).

$$D(x_i, x_j) = (\sum_{l=1}^{d} \sqrt{|x_{il} - x_{jl}|})^2 \tag{1.1}$$

En la ecuación(1.1) x_i y x_j son dos objetos geométricos dispuestos en un espacio y d es la dimensión de dicho espacio.

Como puede apreciarse, para un único punto de consulta el algoritmo descrito anteriormente y ejemplificado en la Figura(1.3) es de hecho óptimo. En la figura se emplea dos como parámetro indicador de la cantidad mínima de vecinos a encontrar. Como resultado los puntos retornados por la consulta son A y B por ser los más cercanos a C. Sin embargo para n puntos de consulta para los que se desea calcular el vecino más cercano la complejidad temporal de este algoritmo es de $O(n^2)$ (Chen, 1996).

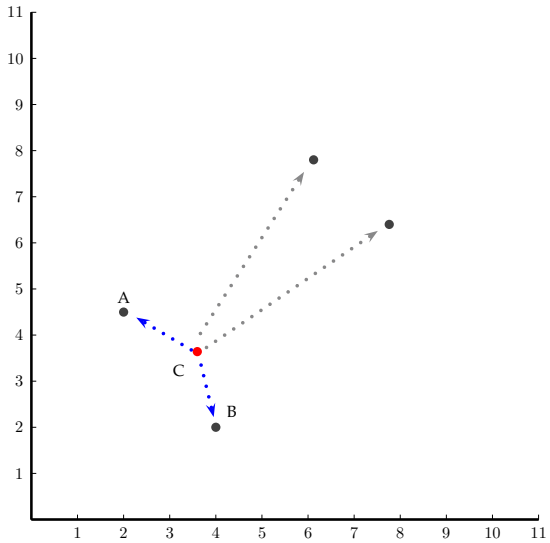


Figura 1.3: *Búsqueda en el espacio de los vecinos cercanos a un punto de consulta.*

[3] *Clustering* se refiere a un grupo de algoritmos para clasificar objetos en clases o agrupaciones (Xu y Wunsch, 2008). El término en español sería agrupamiento, sin embargo se utiliza el término en inglés por considerarse más autorizado.

1.3.2. Búsquedas regionales

En su forma más general, las búsquedas regionales o por rango consisten en procesar una serie de objetos en función de determinar cuáles de ellos se intersecan con un objeto de consulta determinado (De Berg et al., 2000). Por ejemplo dado una serie de muestras geológicas puntuales como las de la Figura(1.1b) se desea conocer cuáles de estas se encuentran contenidas en una esfera dada. Existen diferentes variaciones de este problema determinado fundamentalmente por los tipos de datos sobre los cuales se efectúan las consultas espaciales y por el tipo de objeto de consulta:

- **Tipo de datos a buscar**, los algoritmos deben variar en función de si los objetos son puntos, segmentos, cajas, polígonos, etc.

- **Tipo objeto de consulta**, los algoritmos deben variar en función de si el objeto es alguna especie de ortoedro en cuyo caso se estaría en presencia de una *búsqueda ortogonal* y si fuera una esfera se estaría en presencia de una *búsqueda radial*. En la Figura(1.4) se muestra un grupo de puntos en un espacio bidimensional representando a una búsqueda rectangular análoga a una búsqueda ortogonal en un espacio tridimensional. Como puede apreciarse los puntos señalados con cruces deben ser devueltos finalmente por la consulta.

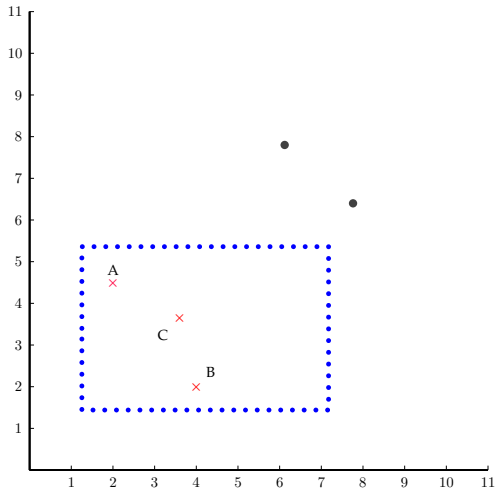


Figura 1.4: *Representación gráfica de una búsqueda ortogonal en un espacio tridimensional.*

El problema de la búsqueda por rango puede ser definido como un problema de intersección geométrica. Éstos pueden expresarse como encontrar aquellas porciones del espacio en las que dos objetos coinciden. Los problemas de intersección y sus variantes surgen en muchas disciplinas tales como el diseño arquitectónico, gráfico por computadoras o reconocimiento de patrones (Chen, 1996).

Una revisión de la literatura especializada muestra que existen dos enfoques principales para disminuir el consumo de tiempo de las búsquedas espaciales: el primero es la computación paralela[4] (Ibaroudene y Acharya, 1995) y el segundo mediante el empleo de los índices espaciales. El empleo de la computación paralela implica dos dificultades fundamentales, la necesidad de contar con hardware que permita ejecutar varias instrucciones simultáneamente y la complejidad inherente a escribir algoritmos concurrentes. Dado esto, la presente investigación se interesa en la utilización de las estructuras de datos espaciales para disminuir el consumo de tiempo de las consultas espaciales.

1.4. Estructuras de datos espaciales

Una estructura de datos espacial organiza una serie de objetos geométricos siguiendo cierto criterio de agrupamiento. Estas estructuras de datos pueden ser utilizadas para acelerar el proceso de consulta sobre un grupo de objetos de forma tal que sea posible conocer en qué posición del espacio se encuentran. La naturaleza propia de las búsquedas espaciales hace que este tipo de estructura de datos sea empleada en una gran variedad de aplicaciones tales como pruebas de oclusión, el cálculo de volumen y la detección de colisiones (Akenine-Möller et al., 2008).

Las estructuras de datos espaciales pueden dividirse en dos clases según su propósito, modelar o indexar. Las estructuras de datos para modelar persiguen como objetivo representar la estructura espacial de cierto conjunto de datos y son muy comunes en el modelado de sólidos y la simulación de fenómenos. Las estructuras de datos para indexar se caracterizan por facilitar la consulta y obtención de datos de forma tal que sea posible obtener rápidamente información asociada.

En los últimos años se ha realizado un cúmulo considerable de esfuerzos en la investigación de las estructuras de datos espaciales para realizar búsquedas espaciales específicas. Esto ha permitido en determinada medida disminuir el consumo de tiempo de dichas consultas ligeramente. En (Chazelle y Welzl, 1989) por ejemplo, se demuestra cómo las búsquedas por rango donde el objeto de consulta es un triángulo pueden realizarse en $O(\sqrt{n} \times \log n)$ pero solo bajo determinadas condiciones preestablecidas. Considerando lo anterior, para la presente investigación solo resultan interesantes aquellas estructuras espaciales de propósito general con la capacidad de resolver consultas de proximidad y búsquedas regionales.

Según (Akenine-Möller et al., 2008) la organización de una estructura de datos espacial es generalmente jerárquica y su construcción es una tarea costosa en términos computacionales, por lo que usualmente se lleva a cabo como un preproceso. Por lo general, se construye un índice de los objetos geométricos de una escena y posteriormente pueden realizarse varias consultas espaciales sin necesidad de reconstruirlo. En algunas aplicaciones es posible incluso realizar modificaciones incrementales. Algunas estructuras de datos

[4]La computación paralela es una forma de cómputo en la que muchas instrucciones se ejecutan de forma simultánea, operando sobre el principio de que problemas grandes pueden ser divididos en instancias más pequeñas y solubles (Almasi y Gottlieb, 1988).

espaciales son los *Bounding Volume Hierarchies*[5] (BVHs), *Binary Space Partitioning*[6] (BSP) y los *Octree*[7].

Un *Bounding Volume (BV)* o volumen de restricción es un volumen que contiene un conjunto de objetos dentro de si. El principio de un BV es que debe ser más fácil tratar un único volumen en el espacio que un conjunto de objetos. Algunos ejemplos son los *Axis Aligned Bounding Boxes*[8] (AABB) y los *Oriented Bounding Boxes*[9] (OBB). Este tipo de estructuras es utilizada mayormente en aplicaciones para el renderizado en tiempo real (Akenine-Möller et al., 2008). En la Figura(1.5) puede apreciarse cómo un grupo de objetos geométricos son organizados jerárquicamente en un BVH. En la sección izquierda existen seis objetos encerrados en esferas (BV). Estas esferas son a su vez encerradas en otras esferas de mayor tamaño. En la sección derecha se muestra el BVH resultante. Nótese como el BV de la raíz encierra toda la escena.

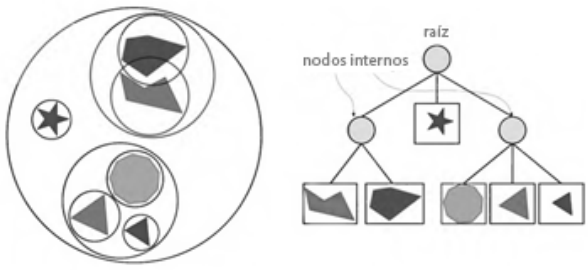


Figura 1.5: *Proceso de indexación espacial mediante un VBH, tomada de (Akenine-Möller et al., 2008).*

Los VBHs son excelentes para ejecutar determinados tipos de consultas como por ejemplo retornar la primera primitiva que es interceptada por un rayo trazado en el espacio. También son ampliamente utilizados en la construcción dinámica de espacios donde los objetos no poseen una misma posición relativa sino que se mueven en éste. Sin embargo no son muy eficientes resolviendo problemas de proximidad debido a que no realizan ningún tipo ordenación de las primitivas ubicadas en el espacio y por lo general no son muy eficientes realizando búsquedas espaciales.

Los BSP en cambio son creados al usar un plano para dividir el espacio en dos para luego ordenar los objetos geométricos que se encuentran a ambos lados de dicho plano. Los BSP pueden proveer una representación computacional del espacio que actúa como una estructura de búsqueda y como una representación geométrica del mismo. En el proceso de construcción de un BSP primero se computan las relaciones espaciales entre los objetos que se encuentran en el espacio para luego codificar dichas relaciones en un árbol binario (Mehta y Sahni, 2004).

[5] *Bounding Volume Hierarchies*: El término en español sería Jerarquía de Volúmenes de Restricción, sin embargo se utiliza el término en inglés por considerarse más autorizado.

[6] *Binary Space Partitioning*: El término en español sería Particionado Binario del Espacio, sin embargo se utiliza el término en inglés por considerarse más autorizado.

[7] *Octree*: El término en español sería árbol octal, sin embargo se utiliza el término en inglés por considerarse más autorizado.

[8] *Axis Aligned Bounding Boxes*: Hace referencia a un volumen de restricción que se encuentra alineado respecto a los tres ejes coordenados del sistema cartesiano.

[9] *Oriented Bounding Boxes*: Hace referencia a un volumen de restricción que posee una orientación arbitraria respecto al espacio, no necesariamente en el sentido de los ejes coordenados del sistema cartesiano.

Existen dos versiones del BSP, los *Axis Aligned* BSP y los *Polygon Aligned* BSP. Para ambos casos el proceso de construcción es siempre el mismo; primero se genera una línea, plano o hiperplano en dependencia de la dimensión del espacio en cuestión para dividirlo en dos y posteriormente se organizan las geometrías a un lado u otro, tal como se muestra en la Figura(1.6). En determinadas bibliografías es posible encontrar la denominación de *K-d tree* para los BSP que trabajan en espacios de más de tres dimensiones. A pesar de lo anterior el proceso de construcción de un BSP donde el espacio es subdividido en dos no parece muy natural para subdividir eficientemente un espacio tridimensional.

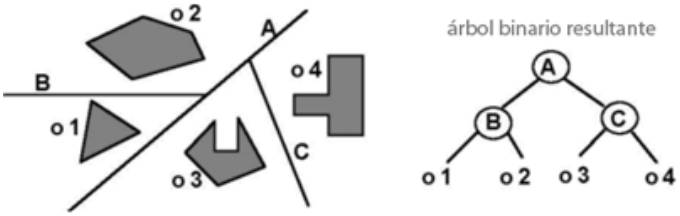


Figura 1.6: *En la sección izquierda se muestra un grupo de objetos que han sido indexados por un BSP. En la sección derecha la estructura jerárquica resultante, tomada de (Mehta y Sahni, 2004).*

1.4.1. Caracterización de los Quadtree y los Octree

Los *Quadtree* son unas estructuras espaciales jerárquicas que se caracterizan por subdividir recursivamente el espacio. El término *Quadtree* se origina por el hecho de indexar objetos geométricos, generalmente en un espacio bidimensional, mediante la división recursiva del espacio empleando separadores paralelos a los ejes de coordenadas. El resultado es una región espacial dividida en 4 subregiones representado por cuatro hijos asociados a un nodo correspondiente a la región original en una estructura jerárquica. Análogamente en un espacio tridimensional, una región es dividida en ocho subregiones usando planos paralelos a los tres ejes coordenados. Como el nodo asociado a la región original posee ocho hijos correspondientes a las 8 subregiones resultantes el término *Octree* es empleado en este caso (Mehta y Sahni, 2004). En la Figura(1.7) se muestra gráficamente un *Octree*.

Existen otras denominaciones para estructuras de datos espaciales similares a los *Quadtree* y los *Octree*. El término *hyperoctree* es empleado para aquellas estructuras de datos espaciales que se especializan en subdividir espacios de elevadas dimensiones. Es también una práctica común utilizar indiscriminadamente el término *Quadtree* para referirse a cualquier estructura de datos espacial a la que le es aplicable los mismos principios que a los *Quadtree*.

Las estructuras de datos de tipo *Octree* fueron propuestas por primera vez por Tanimoto y Jackings (Jackins y Tanimoto, 1980) y estudiadas en profundidad por (Meagher, 1982). Un Octree en su forma más general es una aproximación tridimensional de un objeto por un conjunto de cubos de diferentes tamaños (Yamaguchi et al., 1984a). Estos cubos se encuentran jerárquicamente organizados de tal forma que el

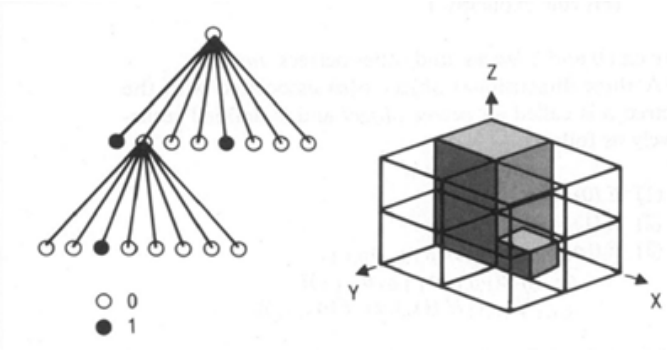


Figura 1.7: *En la sección izquierda se muestra la estructura jerárquica de un Octree y en la sección derecha el correspondiente objeto en un espacio tridimensional, tomada de (Yamaguchi et al., 1984a).*

tiempo de búsqueda puede ser reducido dejando de analizar los pequeños cubos dentro de los de mayor tamaño. El costo para la construcción de un *Octree* es de $O(n \times \log n)$ donde n es la cantidad de objetos geométricos de la escena (Mehta y Sahni, 2004).

Por ejemplo, considérese el problema de encontrar todos los puntos que se encuentran a una distancia determinada de un punto de consulta, o lo que es conocido también como una búsqueda radial o esférica. De no existir ninguna organización en los datos esto requeriría realizar una medición de la distancia entre el punto de consulta y todos los puntos del conjunto de datos. En otro caso, de existir un *Octree* indexando los datos, grandes regiones que se encuentran fuera de la región esférica de consulta pueden ser rápidamente descartadas resultando en menores tiempos de ejecución.

Los *Octree* son ampliamente utilizados por varias razones. En primer lugar proveen un mecanismo para manejar regiones similares de información espacial con un bajo costo de tiempo y memoria. En segundo lugar los conceptos relacionados con árboles y estructuras jerárquicas son muy bien comprendidos y existen muchos trabajos relacionados con esto. Finalmente, muchas operaciones, como la búsqueda del vecino más cercano y muchas otras, se encuentran implementadas en términos de operaciones sobre árboles (Tu et al., 2003). Los *Octree* son ampliamente usados en aplicaciones para la representación de sólidos de alta complejidad, análisis de comportamiento de sistemas en la ingeniería, detección de colisiones y pruebas de oclusión (Moore y Wilhelms, 1988).

Existen diferentes alternativas a la hora de implementar un *Octree*. Estos puede ser implementado empleando referencias mediante punteros o una estructura lineal (Dunstan y Mill, 1989). Los *Octree* Lineales o *Linear Octree* suelen ahorrar espacio de memoria al reducir la cantidad de nodos que poseen, sin embargo son difíciles de implementar.

De acuerdo a (Akenine-Möller et al., 2008) la utilización de *Octree* en vez de una búsqueda secuencial suele representar una mejora temporal de $O(n)$ a $O(\log_8 n)$ para la mayoría de los problemas de búsqueda espacial, lo que representa también una mejora respecto a los BSP anteriormente enunciados. En la Figura(1.8) se muestra el análisis de la complejidad temporal para una estructura de datos lineal y un *Octree* en líneas discontinuas donde teóricamente, las búsquedas empleando una estructura de datos lineal siempre se comporta peor. El campo de las investigaciones de los *Octree* es uno de los más activos en los últimos

años y actualmente existe mucha documentación y teoría elaborada sobre los mismos.

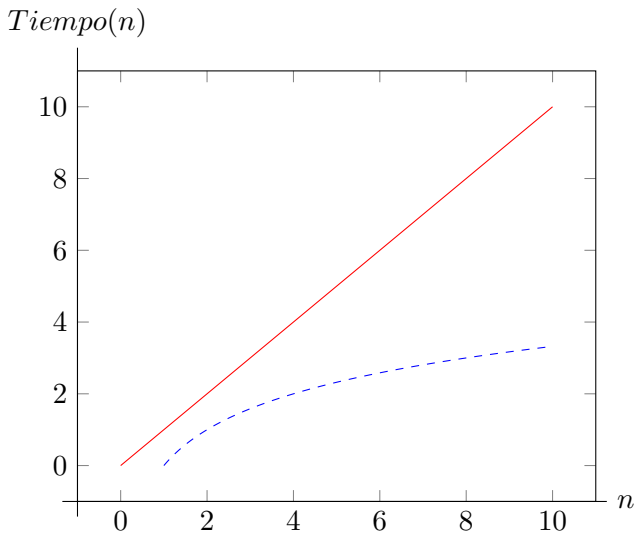


Figura 1.8: *Análisis de la complejidad temporal entre una búsqueda secuencial en linea continua y una búsqueda mediante Octree en linea discontinua.*

1.4.2. Definición de los *Octree*

Un *Octree* es una estructura de datos espacial jerárquica diseñada para indexar objetos geométricos en un espacio tridimensional. Cada nodo del árbol se hace corresponder con un volumen de restricción que encierra un grupo de objetos geométricos. Si no existe ningún objeto geométrico de la escena contenido en un volumen de restricción dado no se hace necesario subdividir el espacio. En cualquier otro caso se comienza un proceso de subdivisión recursiva del espacio siguiendo cierto criterio de indexación.

Según (Yamaguchi et al., 1984a), sea un *Octree* O, S un conjunto de objetos geométricos contenidos dentro del volumen de restricción V correspondiente a O y $P(\{O, S, V\}) \rightarrow 0, 1$ una función que determina si el *Octree* O cumple con un criterio de indexación específico. La definición formal viene dada según la aplicación de las siguientes tres reglas para su construcción:

$$\begin{array}{lcc} 1 & O & \text{si } |S| = 0 \\ 2 & O & \text{si } P(O, S, V) = 1 \\ 3 & \{\{O_1, S_1, V_1\}, ..., \{O_8, S_8, V_8\}\} & \text{en otro caso} \end{array} \tag{1.2}$$

En la anterior definición el proceso de construcción de un *Octree* O se detiene si su volumen de restricción no contiene ningún objeto geométrico o si su volumen de restricción encierra todos los objetos geométricos de la escena y se ha cumplido el criterio de indexación dado. En otro caso se subdivide el espacio en ocho particiones O_i y se le aplica recursivamente las mismas tres reglas definidas anteriormente, recalculando el volumen de restricción correspondiente a cada nodo V_i, así como los objetos geométricos que contienen S_i.

1.4.3. Criterios para la indexación espacial

El proceso de subdivisión del espacio para los *Octree* posee un carácter recursivo y jerárquico. Según (Aronov et al., 2003) para los *Octree* el grado de adaptabilidad para resolver un grupo de problemas depende de una serie de parámetros manualmente seleccionados que controlan cuándo detener el proceso de particionado del espacio. Una mayor o menor cantidad de nodos generados influye en la cantidad de iteraciones necesarias para calcular el resultado de una consulta espacial. Esto se debe a que para un *Octree* el comportamiento de la complejidad temporal es logarítmico mientras se realizan búsquedas en la estructura jerárquica y lineal al realizar búsquedas de los objetos espaciales indexados en un nodo específico. A pesar de esto, el establecimiento de dichos parámetros para diferentes escenarios, específicamente de un umbral de recurrencia(β) que determine la profundidad de la estructura jerárquica, no es una tarea trivial y se ha escrito muy poco sobre esto. Se identifican tres criterios que se corresponden al proceso de definir la función $P(O, S, V)$ dada la definición de *Octree* en el acápite anterior:

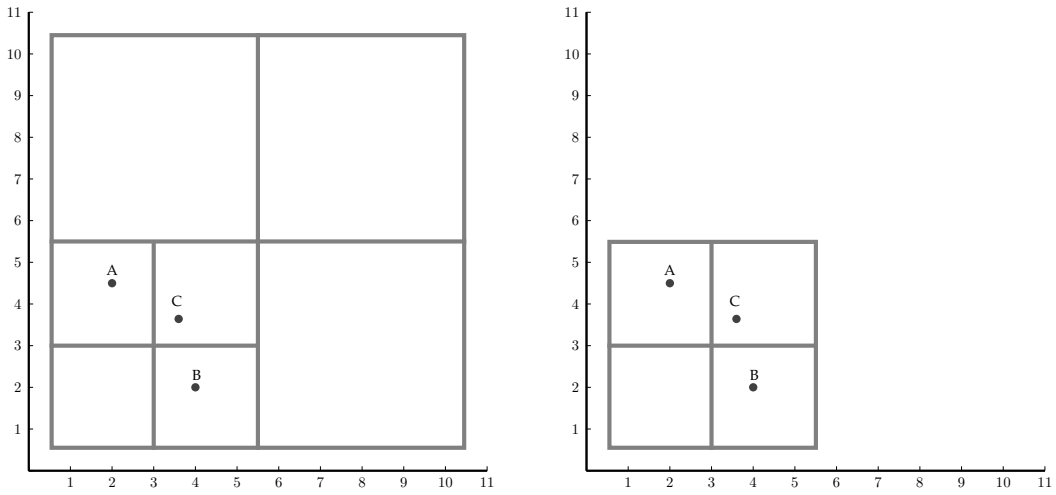
- **Umbral de primitivas geométricas (θ)** por partición del espacio, subdividir mientras cuando $\theta < |S|$.
- **Umbral volumétrico(δ)** para cada partición del espacio, subdividir mientras $\delta < V.volumen$.
- **Nivel de profundidad(β)** en la estructura jerárquica, subdividir mientras $\beta > O.profundidad$.

1.4.4. Establecimiento del volumen de restricción inicial

Durante el proceso de construcción de un *Octree* se establece el criterio para la selección del volumen de restricción que encierra la escena. Este es un parámetro de ajuste muy importante pues puede definir el balance del árbol con repercusiones en el tiempo de ejecución de las búsquedas espaciales. En la Figura(1.9a) se muestra cómo elegir un volumen de restricción arbitrario provoca la creación de un mayor número de particiones del espacio que en la Figura(1.9b). Esto se realiza siguiendo el criterio de subdivisión al sobrepasar un umbral($\theta = 1$) de primitivas geométricas por partición del espacio.

Existen cuatro criterios fundamentales para el establecimiento del volumen de restricción:

- **Ajuste arbitrario** donde las fronteras del volumen de restricción inicial se establecen a partir de criterios que aseguran que los objetos geométricos que componen la escena se encuentran dentro del mismo.
- **Volumen de restricción mínimo** donde las fronteras del volumen de restricción inicial se establecen a partir del volumen más pequeño que encierra los objetos geométricos de la escena.
- **Cálculo del centroide** de los objetos geométricos que conforman la escena. La idea básica tras el uso este método es emplear el centro de los objetos geométricos para lograr una buena distribución de estos en todas las ramas del *Octree*. A pesar de que esta pudiera considerarse una buena alternativa no siempre ofrece buenos resultados.



(a) (b)

Figura 1.9: *Situación de balance al elegir el volumen de restricción inicial en un Octree.*

- **Técnicas heurísticas** que permitan calcular un volumen de restricción que no provoque situaciones anómalas de balance. La idea es utilizar cierto conocimiento a priori de la estructura y el uso que se hará de los datos para establecer un volumen de restricción adecuado.

1.5. Análisis de las soluciones existentes

Actualmente existe un grupo de soluciones con el propósito disminuir el tiempo de ejecución al realizar búsquedas espaciales. Estas soluciones se encuentran disponibles en Internet para su uso en diferentes proyectos y con diferentes propósitos. Para evaluar la factibilidad de su uso, se establece una serie de criterios que permiten realizar un análisis de dichas soluciones. Estos criterios se encuentran listados a continuación:

1. Implementación de búsquedas del vecino más cercano.
2. Implementación de búsquedas por rango.
3. Primitivas geométricas indexadas.
4. Criterios de indexación espacial implementados.
5. Tipo de método para la selección del volumen de restricción inicial.
6. Referencias al nodo padre.

En la Tabla(1.1) se muestran las principales bibliotecas de clases que implementan estructuras de datos *Octree* relacionadas con el autor y disponibilidad de las mismas en Internet. A continuación se realiza un

análisis de las principales soluciones encontradas teniendo en cuenta los criterios definidos. En la Tabla(1.2) puede apreciarse una matriz que relaciona las soluciones estudiadas con los criterios de análisis.

Tabla 1.1: *Soluciones de software que implementan las estructuras de datos Octree analizadas y se encuentran disponibles en la Internet.*

No.	**Autor(es)**	**Disponibilidad**
1	Paul Nettle	http://www.flipcode.com/archives/ Octree_Implementation.shtml
2	Lynn Jones	http://akbar.marlboro.edu/mahoney/support/alg/ alg/node41.html
3	VTK	http://www.vtk.org/doc/nightly/html/ classvtkHyperOctree.html
4	Tero Karras, Samuli Laine	https://code.google.com/p/efficient-sparse-voxel-octrees
5	Harrison Ainsworth	https://github.com/hxa7241/octree-cpp
6	Simon Perreault	http://nomis80.org/code/octree.html

Tabla 1.2: *Principales deficiencias encontradas en las soluciones presentadas para el problema de disminuir el tiempo de ejecución de las búsquedas espaciales.*

S/C	**1**	**2**	**3**	**4**	**5**	**6**
1	No	No	Puntos	Umbral de objetos geométricos, no permite modificar el umbral	Volumen de restricción mínimo	No
2	No	No	Puntos	Umbral volumétrico, no permite modificar el umbral	Ajuste arbitrario	Sí
3	No	No	Puntos	Umbral de objetos geométricos, Umbral volumétrico	Ajuste arbotrario	No
4	No	No	Puntos	Umbral de objetos geométricos	Ajuste arbitrario	No
5	No	Sí	Puntos	Umbral volumétrico	Ajuste arbitrario	No
6	No	No	Puntos	Umbral de objetos geométricos	Volumen de restricción mínimo	No

Las soluciones analizadas no permiten realizar búsquedas del tipo vecino más cercano y la mayoría no posee implementadas rutinas de búsqueda regional. Ninguna de las soluciones vistas permite indexar objetos geométricos como líneas o triángulos, así como tampoco implementan los criterios de indexación de forma que sea posible establecer arbitrariamente los umbrales para detener el proceso de particionado el espacio. De la misma forma no poseen mecanismos para el establecimiento del volumen de restricción de los datos, pudiendo dar lugar a anomalías de desbalance como las vistas en el Epígrafe (1.4.4). Algunas de estas bibliotecas de clases ofrecen muy pocas posibilidades de escalabilidad, lo que dificulta agregar nuevas funcionalidades debido a un pobre diseño de clases o escasa documentación que clarifique la codificación.

1.6. Conclusiones parciales del capítulo

Se analizaron un grupo de estructuras de datos espaciales con el propósito de disminuir el tiempo de ejecución de búsquedas espaciales. Entre estas, las estructuras de datos *Octree* son las que poseen las mejores características para disminuir el tiempo de ejecución de las búsquedas del vecino más cercano, las búsquedas ortogonales y las búsquedas radiales. Desde el punto de vista teórico, es la que posee menor complejidad temporal entre las estructuras de datos analizadas. En la actualidad se ha escrito muy poco sobre los criterios de indexación y umbrales de recursividad para estas estructuras de datos, existiendo una rama de investigación incipiente en este sentido.

Se investigó un grupo de bibliotecas de clases cuyo propósito es disminuir el tiempo de ejecución de las búsquedas espaciales. Estas soluciones analizadas no poseen implementadas rutinas de búsqueda espacial como las búsquedas del vecino más cercano o búsquedas regionales. A su vez, es poco común que dichas soluciones permitan indexar objetos geométricos como líneas y polígonos necesarios para construir índices sobre mallas poligonales. Finalmente, algunas de estas bibliotecas de clases ofrecen muy pocas posibilidades de escalabilidad, lo que dificulta agregar nuevas funcionalidades debido a un pobre diseño de clases o escasa documentación que clarifique la codificación.

Capítulo 2

Propuesta de solución para disminuir el tiempo de ejecución de las búsquedas espaciales

En el siguiente capítulo se analizan un grupo de consideraciones técnicas y de implementación referentes a la propuesta de solución para el problema de disminuir la complejidad temporal de las búsquedas espaciales. La propuesta de solución se encuentra fundamentada sobre la base de los procesos de indexación y descomposición espacial abordados en el Capítulo 1 y específicamente en el uso de una estructura de datos llamada *Octree*.

2.1. Descripción general de la propuesta de solución

Con el objetivo de disminuir el tiempo de ejecución de una búsqueda espacial es necesario categorizar los objetos geométricos de acuerdo a cierto criterio que facilite su posterior consulta y localización en el espacio. El proceso de indexación espacial es costoso en términos computacionales y generalmente se realiza como un preproceso. Esto significa que los datos son indexados y posteriormente se pueden realizar múltiples búsquedas empleando la misma indexación.

La solución propuesta es una biblioteca de clases escrita en el lenguaje de programación C++, específicamente haciendo uso del estándar C++11 (Josuttis, 2012). En la Figura (2.1) se modela el diagrama de clases que representa una abstracción de las principales características implementadas.

En la solución propuesta se hace uso de una variante de *Octree* empleando punteros de acuerdo a la definición dada en el Epígrafe(1.4.2). Cada nodo del árbol resultante se encuentra definido por un volumen de restricción(un ortoedro) que encierra la porción del espacio que le corresponde y un índice de los objetos geométricos contenidos dentro de este. En determinadas búsquedas espaciales, tales como las búsquedas del vecino más cercano, es necesario consultar objetos geométricos que se encuentran en los nodos hermanos. Esta peculiaridad contribuye a que resulte útil poseer también una referencia al nodo padre en función de poder realizar búsquedas hacia los niveles superiores en la estructura jerárquica.

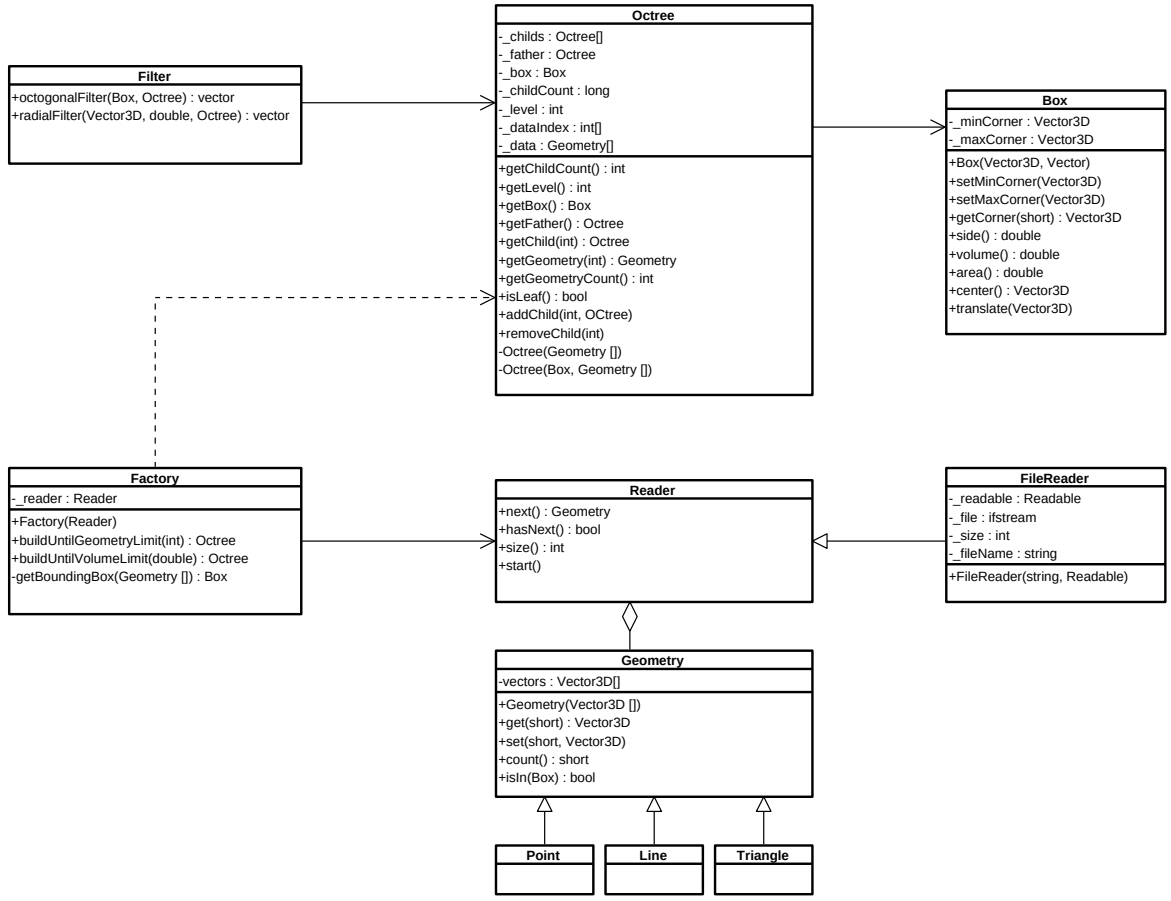
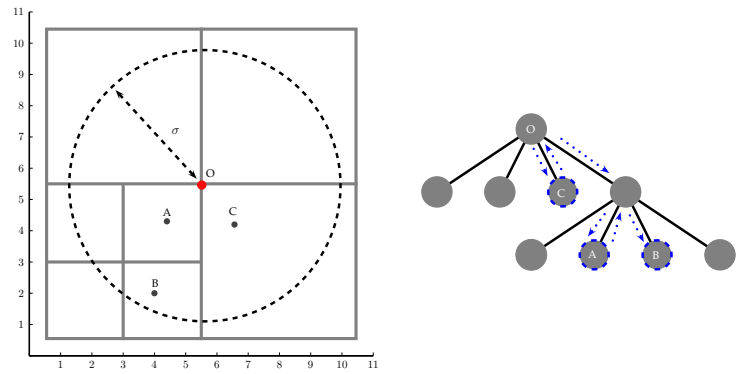


Figura 2.1: *Diagrama de clases de la solución propuesta.*

En la Figura(2.2a) se muestra un grupo de puntos en un espacio bidimensional que han sido indexados mediante un *Quadtree* con propósitos ilustrativos. La indexación espacial se ha realizado siguiendo un criterio de subdivisión donde cada partición del espacio debe poseer solo un objeto geométrico. Al realizar una búsqueda del vecino más cercano a un punto de consulta O como la que se grafica líneas discontinuas, ésta puede comenzar desde el nodo raíz hasta el nodo C. Luego mediante la referencia del nodo padre subir un nivel y procesar los siguientes objetos geométricos, en este caso A y B tal como puede apreciarse en la Figura(2.2b) indicado con flechas de líneas de puntos. La importancia del nodo padre se deriva de la posibilidad de poder realizar el cálculo de la distancia del punto de consulta a los objetos geométricos más cercanos (aquellos contenidos en los nodos hermanos) y abandonar la búsqueda en cierta dirección cuando la distancia sea superior al umbral σ especificado.

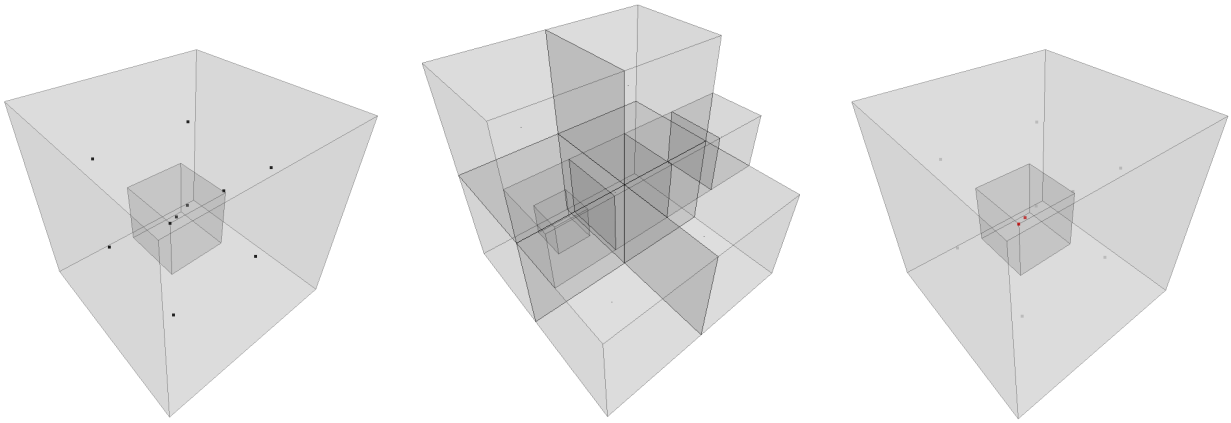
En la Figura(2.3) se presenta una vista de los procesos de indexación espacial y búsqueda espacial. En la Figura(2.3a) se muestra una nube de puntos sobre la cual se dibuja un volumen de búsqueda. Esta



(a) (b)

Figura 2.2: *Descripción de una búsqueda radial.*

nube de puntos es empleada como entrada de un proceso de indexación espacial utilizando el criterio de indexación de 2 objetos geométricos por subdivisión del espacio. El resultado se muestra en la Figura(2.3b). En la Figura(2.3c) se muestra cómo los puntos contenidos dentro del volumen de búsqueda especificado son seleccionados correctamente.



(a) (b) (c)

Figura 2.3: *Proceso de indexación y búsqueda espacial de una nube de puntos simple.*

2.2. Proceso de indexación espacial

El proceso de indexación espacial define el método empleado para categorizar los objetos espaciales de forma tal que puedan ser consultados rápidamente. El proceso de indexación espacial posee un grupo de elementos de entrada y un grupo de elementos de salida al considerarse como un algoritmo. Las entradas de

un algoritmo de indexación espacial pueden ser múltiples, sin embargo en esta investigación la indexación espacial se realiza sobre nubes de puntos y mallas poligonales.

Dado una nube de puntos o una malla poligonal se aplica un método de indexación específico. La naturaleza del proceso de indexación de un *Octree* consiste en subdividir recursivamente el espacio y asociar los objetos geométricos contenidos a su respectiva porción del espacio. Cada porción del espacio queda definida dentro de un volumen de restricción descrito por un ortoedro. Esto determina que el proceso sea regular y que cada objeto geométrico quede finalmente indexado.

Se implementaron diferentes criterios para guiar el proceso de descomposición espacial y cada uno de estos criterios especifica hasta qué punto el espacio es subdividido. Estos fueron abordados en el Epígrafe (1.4.3).

Dada la naturaleza eminentemente recursiva del proceso de indexación espacial descrito, el resultado final o salida de dicho algoritmo es una estructura de datos jerárquica llamada *Octree*. El Algoritmo (1) detalla los pasos de este proceso.

Algorithm 1 Proceso de indexación espacial.

```
1:  function IndexProcces(O : Octree, V : Volume, S : Geometric Object List)
2:      O ← {S, V};
3:      if IndexCriteria(O, S, V) then
4:          for i = 1 : 8 do
5:              O_i : Octree, S_i : Geometric Object List, V_i : Volume;
6:              V_i ← Subdivide(i, V);
7:              S_i ← Intersection(S, V);
8:              IndexProcces(O_i, V_i, S_i)
9:          end for
10:     end if
11: end function
```

El Algoritmo (1) recibe como parámetro una instancia del *Octree* O a indexar, el volumen de restricción V que le corresponde y una lista de los objetos geométricos S que se encuentran contenidos dentro de V. En la instrucción 2 se relaciona el nodo O con su volumen de restricción y los objetos geométricos contenidos dentro del mismo. Luego se evalúa el cumplimiento del criterio de descomposición espacial y de ser necesario, el volumen de restricción del *Octree* en cuestión es subdividido en ocho sub-volúmenes V_i regulares y que en conjunto abarcan exhaustivamente el espacio contenido. Dichos sub-volúmenes son asignados, a los ocho *Octrees* hijos O_i del *Octree* original, y los objetos geométricos de la escena se reorganizan en sus respectivos sub-volúmenes mediante una función de intersección espacial. Este proceso se realiza recursivamente hasta que el criterio de descomposición espacial así lo permita.

El Algoritmo (2) se emplea para calcular el volumen de restricción que encierra los objetos de la escena empleando el método del centroide de los datos. El algoritmo recibe como parámetro una lista S con los puntos de una escena con el objetivo de determinar un volumen de restricción mínimo que los contenga. En la instrucción 3, se calcula el centroide c de los datos garantizándose que el *Octree* resultante se encuentre lo más balanceado posible. Posteriormente se calcula la distancia máxima d del centroide al resto de los objetos geométricos. Finalmente se devuelve un volumen de restricción determinado por

$P_{min} = \{C_x - d, C_y - d, C_z - d\}, P_{max} = \{C_x + d, C_y + d, C_z + d\}$ dos vértices que definen la diagonal de un ortoedro V cuyo origen es el centroide de los datos y radio d.

Algorithm 2 Establecimiento del volumen de restricción inicial para el proceso de indexación espacial.

```
1: function CalculateBoundingVolume(S: Geometric Object List)
2:     V : Volume;
3:     c ← (∑_{q∈S} q_i)/|S|;
4:     d = 0;
5:     for all q ∈ S do
6:         if EuclideanDistance(q, c) > d then
7:             d ←EuclideanDistance(q, c);
8:         end if
9:     end for
10:    Calculate V;
11:    return V;
12: end function
```

2.3. Proceso de búsqueda espacial

El proceso de búsqueda espacial, define los métodos empleados para obtener un grupo de objetos ubicados en el espacio que cumplen con determinadas restricciones geométricas y espaciales. Visto como un algoritmo, el proceso de búsqueda espacial puede ser descrito a partir de entradas y salidas. En esta investigación las entradas de cualquier algoritmo de búsqueda espacial es un *Octree* resultado de un proceso de indexación espacial de los objetos de la escena. Una vez realizada la búsqueda espacial, el resultado es un conjunto de objetos geométricos que cumplen con ciertas restricciones. En la solución propuesta se implementan búsquedas del tipo vecino más cercano y búsquedas regionales ortogonales y radiales.

2.3.1. Búsquedas del vecino más cercano

La definición de la búsqueda del vecino más cercano fue dada en el Capítulo 1. En el presente trabajo se implementa una variante de este tipo de búsqueda. Esta variante puede ser enunciada como encontrar aquellos pares de puntos q_i y q_j cuya distancia euclidiana d sea menor que σ, tal que σ es un umbral de distancia máxima. Este tipo de búsqueda es particularmente útil y es empleado en una gran cantidad de problemas como por ejemplo aquellos en los que dos puntos se encuentran tan cerca entre ellos que pueden ser considerados como uno solo.

Para este tipo de búsqueda resulta ventajoso emplear el criterio de subdivisión hasta alcanzar cierto umbral volumétrico(δ) donde $\delta = \sigma^3$. Posteriormente solo es necesario calcular d en aquellos pares de puntos que se encuentran dentro de un mismo octante y octantes adyacentes del *Octree* resultante. En la Figura (2.4) se ilustra una búsqueda similar en un espacio bidimensional, los puntos devueltos por la consulta espacial son aquellas parejas de puntos contenidos dentro de un mismo cuadrante, o en los cuadrantes vecinos.

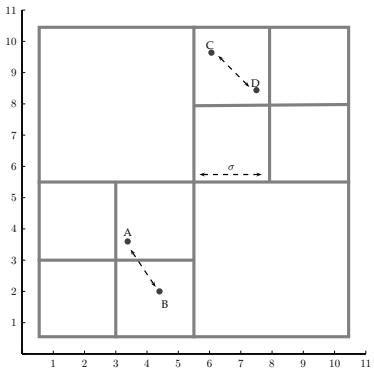


Figura 2.4: *Variante de búsqueda del vecino más cercano.*

2.3.2. Búsquedas regionales

La definición de las búsquedas ortogonales, las búsquedas radiales o esféricas y las búsquedas elípticas viene dada por la naturaleza de la entidad volumétrica empleada para definir las fronteras de la búsqueda. Siendo estas un ortoedro, una esfera y un elipse respectivamente. En sentido general las búsquedas regionales pueden ser enunciadas como el proceso de encontrar todos los objetos geométricos q_i que cumplen con la restricción enunciada en la ecuación 2.1:

$$q_i \in V \tag{2.1}$$

Donde V es un volumen de búsqueda dado, sea un ortoedro, una esfera, un elipse o cualquier otro. El Algoritmo (3) describe el proceso de búsqueda espacial sobre un *Octree* para cualquiera de estos volúmenes de búsqueda.

Algorithm 3 Proceso de búsqueda espacial general.

```
1: function SpatialSearch(V: Search volume, O : Octree)
2:     R: Geometric Objects List;
3:     if VolumeIntersection(V, O.Volume) = True then
4:         if O.IsLeaf = True then
5:             R ← R+ Intersecion(O.S, V);
6:         else
7:             for i = 1..8 do
8:                 R ← R+ SpatialSearch(V, O.O_i);
9:             end for
10:        end if
11:    end if
12:    return R;
13: end function
```

El Algoritmo(3) recibe como parámetro un volumen de búsqueda V y un *Octree* O que es el resultado del proceso de indexación espacial referido en el Algoritmo(1). El procedimiento primeramente determina

si el volumen de restricción del O se interseca con el volumen de búsqueda. De no cumplirse esta condición no se realiza ninguna computación posterior y se infiere que el O no posee objetos geométricos que se intersecan con V. De lo contrario, se determina si O es hoja, en cuyo caso se calcula cuáles de los objetos geométricos contenidos dentro de su volumen de restricción se intersecan con V. Si el *Octree* en cuestión no es hoja, se repite el mismo procedimiento de manera recursiva para cada uno de sus hijos.

Durante el proceso de indexación espacial y búsqueda espacial se realizan pruebas de intersección entre los objetos geométricos de la escena y un grupo de entidades volumétricas[1]. Realizar este tipo de pruebas de intersección aparentemente es una tarea trivial. En la Figura(2.5) existen situaciones análogas en un espacio bidimensional donde las operaciones de intersección espacial son difíciles de efectuar como el caso en el que los puntos de un segmento no se encuentran contenidos dentro de un cuadrado.

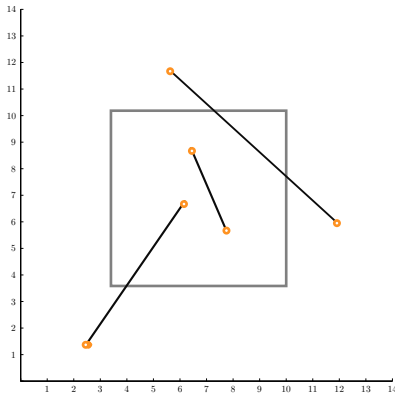


Figura 2.5: *Pruebas de intersección de segmentos con un cuadrado.*

2.4. Conclusiones parciales del capítulo

La implementación de un *Octree* empleando punteros con referencias al nodo padre en la estructura jerárquica, permite disminuir la complejidad temporal de las búsquedas espaciales donde es necesario conocer los nodos adyacentes, como es el caso de la búsqueda del vecino más cercano.

Los algoritmos empleados para la indexación espacial y búsqueda espacial poseen una complejidad temporal de $n \times \log n$ y tienen la ventaja de no tener que recorrer todo el espacio de búsqueda, lo que representa una mejora respecto a la complejidad temporal de una búsqueda espacial empleando una estructura de datos lineal.

[1]Entidad volumétrica: Objeto espacial que por sus características geométricas posee volumen definido.

Capítulo 3

Pruebas y resultados de la estructura de datos *Octree*

En este capítulo se describen los resultados alcanzados por la solución de *Octree* propuesta como solución al problema de disminuir el tiempo de ejecución de las búsquedas espaciales. Para esto se diseñó una serie de pruebas con el fin de medir la demora de realizar búsquedas espaciales dado cierto volumen de datos. Realizar pruebas se refiere al hecho de tomar una acción y observar sus consecuencias. Desde el punto de vista científico se realiza una manipulación intencional de un grupo de variables independientes para medir las consecuencias en una o más variables dependientes en una situación de control.

3.1. Descripción de las pruebas realizadas

Las variables independientes son aquellas que pueden ser manipuladas directamente por el observador de un fenómeno dado. En las pruebas llevadas a cabo en la presente investigación se emplean las siguientes variables independientes:

- Estructura de datos utilizadas durante las búsquedas espaciales.

Las variables dependientes son aquellas que poseen una estrecha relación con las variables independientes ya que su comportamiento se ve afectado por los valores que estas últimas instancian. A continuación se enuncian las variables dependientes identificadas que son de interés para la presente investigación:

- Tiempo de ejecución de las búsquedas espaciales.

La estructura de datos *Octree* descrita en el Capítulo 2 se encuentra implementada en el lenguaje de programación C++, específicamente haciendo uso del estándar C++11 (Josuttis, 2012). Los criterios de selección del lenguaje de programación son los siguientes:

- Es el lenguaje de programación utilizado en la implementación de Syam.

- No es interpretado por lo que resulta más fácil la medición de los tiempos de ejecución ya que no depende de intérpretes o máquinas virtuales.

Debido a que las pruebas son sensibles al planificador de procesos del sistema operativo sobre el cual se realizan, es necesario definir un sistema operativo homogéneo para todos los ensayos. Esto se realizará sobre el sistema operativo Xubuntu 14.10. Se utilizan las siguientes configuraciones de *hardware* seleccionadas por considerarse comunes en los entornos computacionales actuales de acuerdo a (Valle, 2009):

- Celeron Intel Dual Core a 1.66 GHz, 4Gb de RAM.

Los datos de entrada oscilan entre mil y 20 mil objetos geométricos que se consideran pequeñas cantidades (Valle, 2009). Los datos han sido dispuestos aleatoriamente empleando una distribución normal con media aritmética de 500 mil y desviación típica de 200 mil. Para medir los tiempos de ejecución de las búsquedas espaciales se repiten las mismas pruebas varias veces, unas cien mil iteraciones, calculando la demora promedio de todas. Esto hace que se minimice el error cometido en la medición de dichos tiempos de ejecución. Para medir el consumo de memoria física al emplear las distintas estructuras de datos se calcula la cantidad de *bytes* utilizadas por estas.

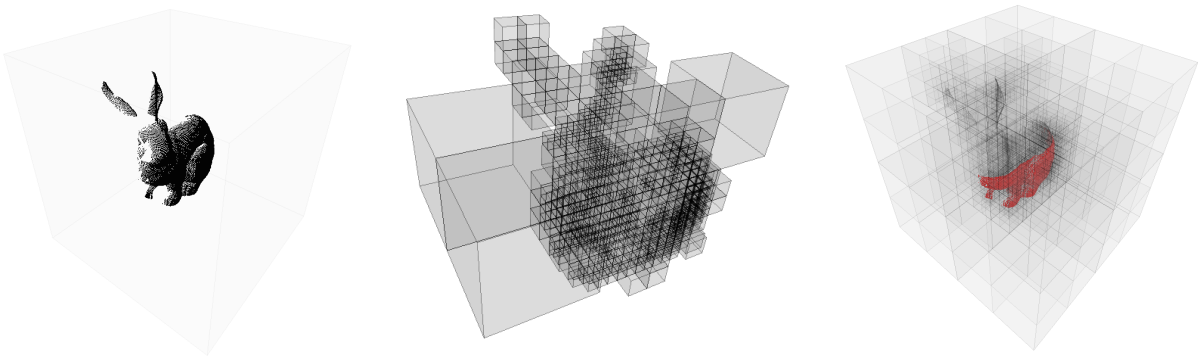
3.2. Resultados de las pruebas realizadas

3.2.1. Visualización tridimensional de la indexación espacial y búsqueda espacial

En la siguiente sección se realiza una prueba de visualización tridimensional de la indexación espacial y la búsqueda espacial. Para esto se emplea una nube de puntos escaneada en la Universidad de Stanford consistente en 362 272 puntos. El objetivo de la prueba es constatar visualmente que los objetos geométricos dispuestos en el escenario de prueba queden correctamente indexados en la subregión espacial correspondiente y que sean adecuadamente devueltos en una posterior consulta espacial.

En la Figura(3.1a) puede apreciarse una representación tridimensional de la nube de puntos antes mencionada así como un volumen de búsqueda ortogonal. Luego se realiza un proceso de indexación espacial empleando el criterio de subdivisión del espacio basado en un umbral (θ) máximo de objetos geométricos por partición del espacio siendo $\theta = 200$. El resultado de dicho proceso de indexación espacial puede verse en la Figura(3.1b). Finalmente los puntos contenidos dentro del volumen de búsqueda son devueltos por la consulta espacial y pueden apreciarse en la Figura(3.1c) representados en color rojo.

Los resultados de la prueba muestran cómo dado un grupo de objetos geométricos dispuestos en un escenario de prueba, éstos son correctamente asignados a una subregión espacial correspondiente a un nodo en el árbol jerárquico resultante. Esto posibilita que a la hora de realizar una búsqueda espacial, los puntos consultados sean devueltos por dicha consulta espacial.



(a) (b) (c)

Figura 3.1: *Proceso de indexación y búsqueda espacial de una nube de puntos.*

3.2.2. Medición del consumo de tiempo de una búsqueda espacial con *Octree* respecto a búsqueda espacial con estructura de datos lineal

La realización de esta prueba tiene como objetivo constatar disminución del consumo de tiempo de las búsquedas espaciales realizadas en la estructura de datos *Octree* respecto a las búsquedas espaciales realizadas en estructuras de datos lineales. Para esto se configura la estructura de datos *Octree* con un grupo de parámetros arbitrarios.

Se emplea el método de selección del volumen de restricción inicial basado en el centroide de los datos. Se realizan mediciones empleando el criterio de indexación espacial estableciendo un umbral (θ) de cien, quinientos y mil objetos geométricos por octante y el criterio de indexación espacial estableciendo un umbral (δ) de 10^{14}, 10^{16} y 10^{18} unidades cúbicas como volumen mínimo de los octantes.

La configuración y distribución de los datos de prueba fueron establecidos en el acápite inicial, empleando muestras en el rango de mil hasta 20 mil puntos. En todos los casos se realizará una búsqueda espacial ortogonal como la descrita en el Capítulo 2; primeramente empleando una estructura de datos lineal y luego una estructura de datos *Octree*. En la Tabla(3.1) se listan los resultados obtenidos de estas mediciones.

Para medir si la estructura de datos propuesta disminuye el consumo de tiempo de las búsquedas espaciales es necesario definir una cota mínima de eficiencia. Para esto se emplea el consumo de tiempo resultante al realizar búsquedas espaciales mediante una estructura de datos lineal en la cual las consultas sobre los objetos geométricos se hagan de forma secuencial. En el Algoritmo(4) se especifica el proceso de búsqueda espacial sobre una estructura de datos lineal.

En la Figura(3.2) se muestran los resultados de medir el consumo de tiempo de una búsqueda ortogonal empleando distintos volúmenes de datos. Puede apreciarse cómo el uso de distintos umbrales para el criterio de indexación espacial de acuerdo a cantidad mínima de objetos geométricos por octante, arroja diferentes niveles de consumo de tiempo. En línea de triángulos se ofrece el consumo de tiempo al emplear una estructura de datos lineal. En la gráfica, el eje de las abscisas presenta un escalado de 1×1000 definiendo

Algorithm 4 Proceso de búsqueda espacial en una estructura de datos lineal.

```
1: function spatialLinearSearch(V : Search Volume, LG: Geometric Objects List)
2:     LR: Geometric Objects List;
3:     for all O : LG do
4:         if Intersect(V,O) = True then
5:             LR ← LR + O;
6:         end if
7:     end for
8:     return LR;
9: end function
```

el volumen de datos empleado; en el eje de las ordenadas se mide el consumo de tiempo en milisegundos de la consulta espacial ortogonal.

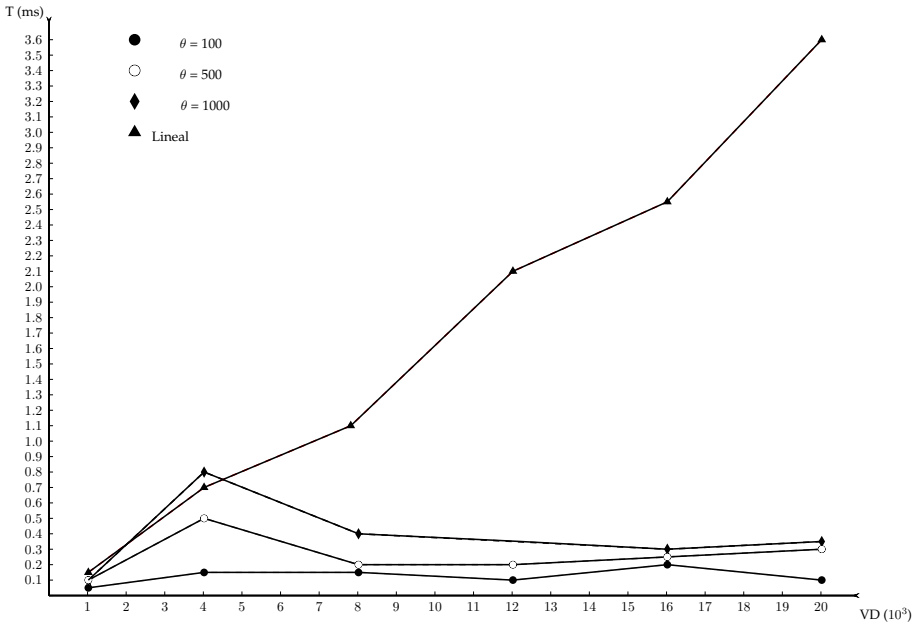


Figura 3.2: *Consumo de tiempo de una consulta espacial ortogonal empleando el criterio de indexación espacial basado en cantidad mínima de objetos geométricos por octante y una estructura lineal.*

En la Figura(3.3) se muestran los resultados de medir el consumo de tiempo de una búsqueda ortogonal empleando distintos volúmenes de datos. Puede apreciarse cómo el uso de distintos umbrales para el criterio de indexación espacial de acuerdo al volumen mínimo del octante, arroja diferentes niveles de consumo de tiempo. En en línea de triángulos se ofrece el consumo de tiempo al emplear una estructura de datos lineal. En la gráfica, el eje de las abscisas presenta un escalado de 1×1000 definiendo el volumen de datos empleado; en el eje de las ordenadas se mide el consumo de tiempo en milisegundos de la consulta espacial ortogonal.

Como puede apreciarse, variaciones en el parámetro referente a la cantidad de objetos geométricos por subregión del espacio, produce variaciones en el consumo de tiempo al emplear distintos volúmenes de

datos. También puede apreciarse cómo las búsquedas espaciales empleando una estructura de datos *Octree*, consumen menor tiempo de ejecución que las búsquedas espaciales realizadas empleando una estructura de datos lineal a partir de cierto volumen de datos. El volumen de datos a partir del cual puede decirse que la búsqueda indexada se realiza en menor tiempo que la búsqueda lineal varía en correspondencia a la cantidad de objetos geométricos por subregión espacial empleada en el proceso de indexación espacial.

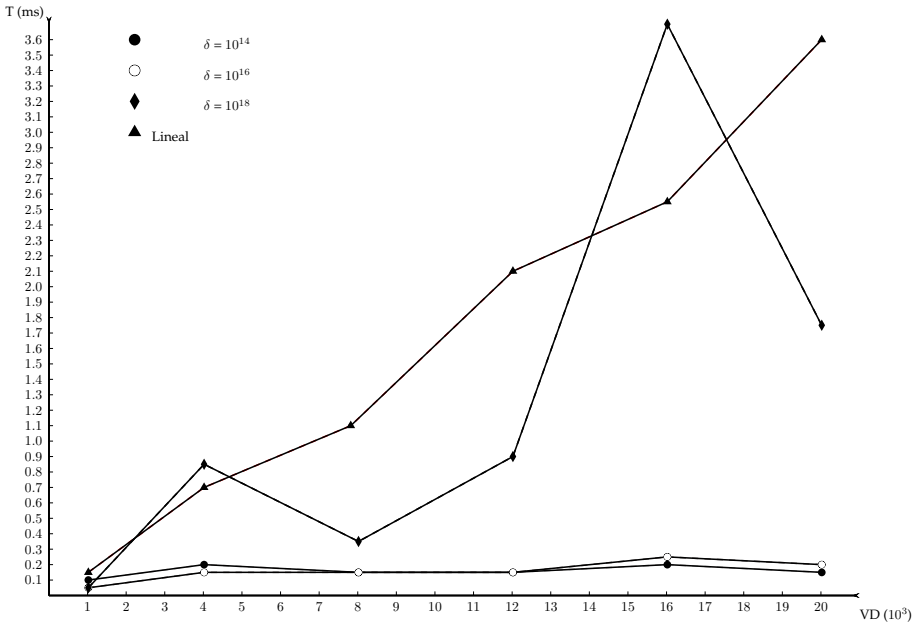


Figura 3.3: *Consumo de tiempo de una consulta espacial ortogonal empleando el criterio de indexación espacial basado en el volumen mínimo de los octantes y una estructura lineal.*

Como puede apreciarse, variaciones en el parámetro referente al volumen mínimo de los octantes, produce variaciones en el consumo de tiempo al emplear distintos volúmenes de datos. También puede apreciarse cómo las búsquedas espaciales empleando una estructura de datos *Octree*, consumen menor tiempo de ejecución que las búsquedas espaciales realizadas empleando una estructura de datos lineal a partir de cierto volumen de datos. El volumen de datos a partir del cual puede decirse que la búsqueda indexada se realiza en menor tiempo que la búsqueda lineal varía en correspondencia al volumen mínimo de los octantes definido en el proceso de indexación espacial. Los resultados obtenidos también muestran cómo existen otros factores además del volumen mínimo de los octantes y la cantidad de datos empleado que influyen en el consumo de tiempo de las búsquedas espaciales. Esto se pone de manifiesto en la oscilación del consumo de tiempo para un $\delta = 10^{18}$. Estos otros factores podrían estar relacionados con la distribución de los objetos geométricos de la escena, su densidad y dispersión.

Tabla 3.1: *Tiempos de ejecución en milisegundos obtenidos para búsquedas ortogonales empleando la estructura de datos Octree y una estructura de datos lineal con diferentes volúmenes de datos y configuraciones.*

	Octree (ms)						**Lineal** (ms)
VD	$\delta = 10^{14}$	$\delta = 10^{16}$	$\delta = 10^{18}$	$\theta = 100$	$\theta = 500$	$\theta = 1000$	
1 000	0.165	0.075	0.078	0.054	0.070	0.072	0.182
4 000	0.230	0.160	0.896	0.155	0.532	0.863	0.711
8 000	0.158	0.139	0.366	0.147	0.240	0.483	1.143
12 000	0.168	0.135	0.931	0.091	0.215	0.209	2.172
16 000	0.233	0.259	3.667	0.193	0.306	0.308	2.900
20 000	0.170	0.204	1.800	0.123	0.298	0.409	3.644

3.2.3. Medición del consumo de memoria física de una estructura de datos *Octree* respecto a una estructura de datos lineal

La realización de esta prueba tiene como objetivo analizar el comportamiento del consumo de memoria física de la estructura de datos *Octree* respecto a una estructura de dato lineal. Para esto se configura la estructura de datos *Octree* con un grupo de parámetros arbitrarios y se mide el consumo de memoria física en *bytes* de ambas estructuras de datos.

Se emplea el método de selección del volumen de restricción inicial basado en el centroide de los datos. Se realizan mediciones empleando el criterio de indexación espacial estableciendo un umbral (θ) de cien, quinientos y mil objetos geométricos por octante y el criterio de indexación espacial estableciendo un umbral (δ) de 10^{14}, 10^{16} y 10^{18} unidades cúbicas como volumen mínimo de los octantes.

La configuración y distribución de los datos de prueba fueron establecidos en el acápite inicial, empleando muestras en el rango de mil hasta 20 mil puntos. En todos los casos se realiza una búsqueda espacial ortogonal como la descrita en el Capítulo 2; primeramente empleando una estructura de datos lineal y luego una estructura de datos *Octree*. En la Tabla(3.2) se listan los resultados obtenidos de estas mediciones.

En la Figura(3.4) se muestran los resultados de medir el consumo de memoria física de una indexación espacial empleando distintos volúmenes de datos. En escala de grises, puede apreciarse cómo el uso de distintos umbrales para el criterio de indexación espacial de acuerdo a cantidad mínima de objetos geométricos por octante, arroja diferentes niveles de consumo de memoria física. En la gráfica, el eje de las abscisas presenta un escalado de 1×1000 definiendo el volumen de datos empleado; en el eje de las ordenadas se mide el consumo de memoria física en *bytes* de la indexación espacial.

En la Figura(3.5) se muestran los resultados de medir el consumo de memoria física de una indexación espacial empleando distintos volúmenes de datos. En escala de grises, puede apreciarse cómo el uso de distintos umbrales para el criterio de indexación espacial de acuerdo al volumen mínimo del octante, arroja diferentes niveles de consumo de memoria física. En la gráfica, el eje de las abscisas presenta un escalado de 1×1000 definiendo el volumen de datos empleado; en el eje de las ordenadas se mide el consumo de memoria física en *bytes* de una indexación espacial.

Como puede apreciarse en la Figura(3.4) y Figura(3.5) al emplear estructuras de datos lineales, el consumo de memoria RAM es menor que al emplear una estructura de datos *Octree* en la generalidad de

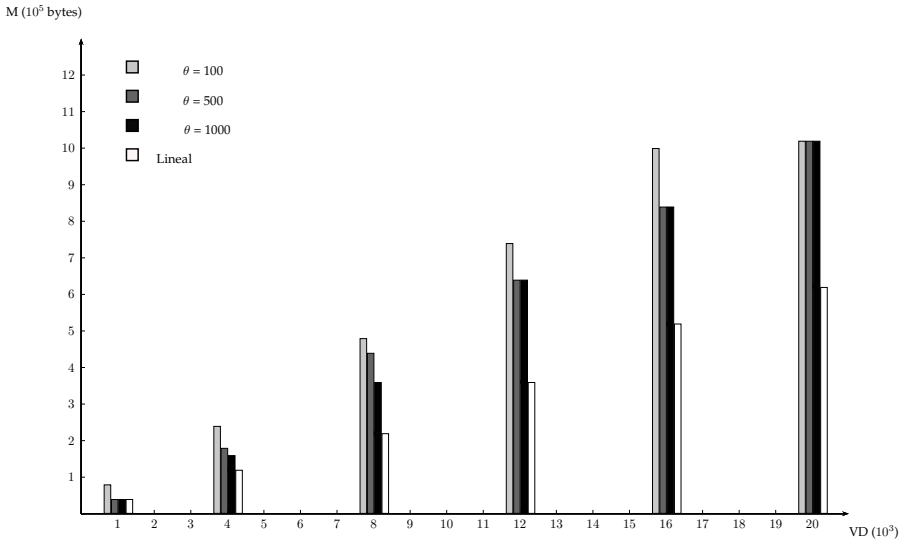


Figura 3.4: *Consumo de memoria física de una indexación espacial empleando el criterio de indexación espacial basado en cantidad mínima de objetos geométricos por octante y una estructura lineal.*

los casos. La variación en los criterios de indexación, así como de sus configuraciones producen distintos niveles de consumo de memoria.

Tabla 3.2: *Consumo de memoria física en bytes obtenido para indexaciones espaciales empleando una estructura de datos Octree y una estructura de datos lineal con diferentes volúmenes de datos y configuraciones.*

	Octree (*bytes*)						**Lineal** (*bytes*)
VD	$\delta = 10^{14}$	$\delta = 10^{16}$	$\delta = 10^{18}$	$\theta = 100$	$\theta = 500$	$\theta = 1000$	
1 000	1464540	195860	56140	70804	41384	41384	32032
4 000	2382396	342452	188140	240544	195196	161384	128032
8 000	3224508	581364	364140	493836	440868	364968	256032
12 000	3738684	782644	540140	756496	644568	644568	384032
16 000	4201788	1010804	716140	1011204	848636	848636	512032
20 000	4777788	1225524	892140	1233000	1070928	1050588	640032

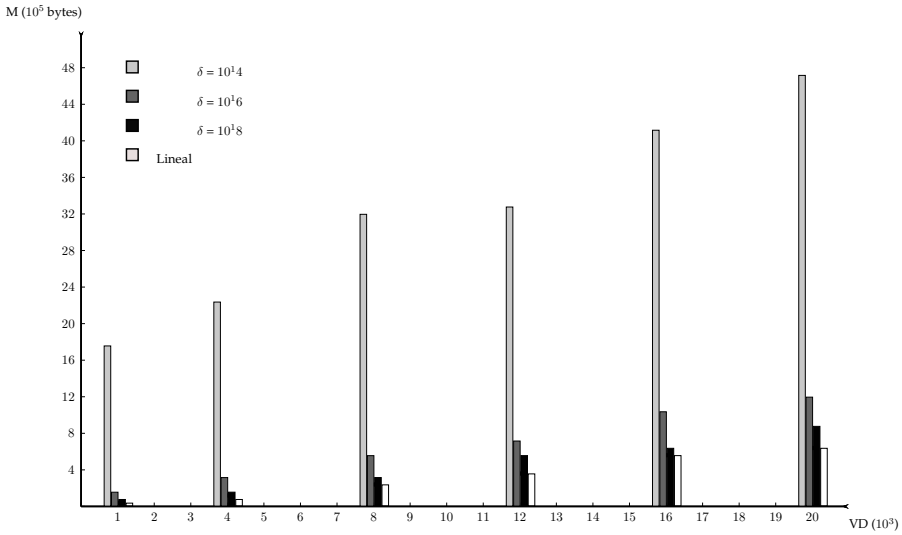


Figura 3.5: *Consumo de memoria física de una indexación espacial empleando el criterio de indexación espacial basado en el volumen mínimo de los octantes y una estructura lineal.*

3.3. Conclusiones parciales del capítulo

Los resultados de las mediciones realizadas muestran, que el tiempo de ejecución al efectuar búsquedas espaciales empleando estructuras de datos lineales, en comparación con un *Octree*, se comporta de acuerdo a los presupuestos teóricos a partir de cierto volumen de datos. Se observa que existe dependencia entre el volumen de datos y el criterio de indexación empleado, así como entre el volumen de datos y otros factores a investigar, evidenciándose en distintos niveles de tiempo de ejecución para cada umbral.

El consumo de memoria RAM empleando una estructura de datos lineal es inferior al obtenido empleando un *Octree*. La variación en los criterios de indexación, así como de sus configuraciones producen distintos niveles de consumo de memoria RAM para un *Octree*.

Conclusiones

La investigación mostró resultados positivos, los cuales pueden referirse así:

- Se investigó un grupo de bibliotecas de clases cuyo propósito es disminuir el tiempo de ejecución de las búsquedas espaciales. Estas soluciones analizadas no poseen implementadas rutinas de búsqueda espacial como las búsquedas del vecino más cercano o búsquedas regionales. A su vez, es poco común que dichas soluciones permitan indexar objetos geométricos como líneas y polígonos necesarios para construir índices sobre mallas poligonales. Finalmente, algunas de estas bibliotecas de clases ofrecen muy pocas posibilidades de escalabilidad, lo que dificulta agregar nuevas funcionalidades debido a un pobre diseño de clases o escasa documentación que clarifique la codificación.
- La implementación de un *Octree* empleando referencias a los nodos padres en la estructura jerárquica en conjunto con el empleo de algoritmos de indexación espacial con complejidad temporal de $n \times \log_8 n$ permitió disminuir la complejidad temporal ofrecida por las estructuras de datos lineales para búsquedas espaciales.
- Los resultados de las mediciones realizadas muestran, que el tiempo de ejecución al efectuar búsquedas espaciales empleando estructuras de datos lineales, en comparación con un *Octree*, se comporta de acuerdo a los presupuestos teóricos a partir de cierto volúmen de datos. Se observa que existe dependencia entre el volumen de datos y el criterio de indexación empleado, así como entre el volúmen de datos y otros factores a investigar, evidenciándose en distintos niveles de tiempo de ejecución para cada umbral.
- El consumo de memoria RAM empleando una estructura de datos lineal es inferior al obtenido empleando un *Octree*. La variación en los criterios de indexación, así como de sus configuraciones producen distintos niveles de consumo de memoria RAM para un *Octree*.

Bibliografía consultada y referencias bibliográficas

Alias Abdul-Rahman y Morakot Pilouk. *Spatial data modelling for 3D GIS*. Springer Science & Business Media, 2007.

Tomas Akenine-Möller, Eric Haines, y Naty Hoffman. *Real-time rendering*. CRC Press, 2008.

George S Almasi y Allan Gottlieb. Highly parallel computing. 1988.

Boris Aronov, Hervé Bronnimann, Allen Y Chang, y Yi-Jen Chiang. Cost-driven octree construction schemes: an experimental study. En *Proceedings of the nineteenth annual symposium on Computational geometry*, págs. 227–236. ACM, 2003.

J Bai, Xuesheng Zhao, y J Chen. Indexing of the discrete global grid using linear quadtree. *Proceedings of the XXXVIth ISPRS*, págs. 267–270, 2005.

Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Lévy, Stephan Bischoff, y Christian Rössl. Geometric modeling based on polygonal meshes. En *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07. ACM, New York, NY, USA, 2007. ISBN 978-1-4503-1823-5. doi:10.1145/1281500.1281640. URL http://doi.acm.org/10.1145/1281500.1281640.

Mario Botsch, Stephan Steinberg, Stephan Bischoff, y Leif Kobbelt. Openmesh-a generic and efficient polygon mesh data structure. 2002.

Henry Ker-Chang Chang, Shing-Hua Liu, y Cheng-Kuan Tso. Two-dimensional template-based encoding for linear quadtree representation. *Photogrammetric engineering and remote sensing*, 63:1275–1284, 1997.

Bernard Chazelle y Emo Welzl. Quasi-optimal range searching in spaces of finite vc-dimension. *Discrete & Computational Geometry*, 4(1):467–489, 1989.

J. Chen. Computational geometry: methods and applications. *Computer Science Department Texas University*, 1996.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, et al. *Introduction to algorithms*, tomo 2. MIT press Cambridge, 2001.

Mark De Berg, Marc Van Kreveld, Mark Overmars, y Otfried Cheong Schwarzkopf. *Computational geometry*. Springer, 2000.

LouisJ Doctor y JG Torborg. Display techniques for octree-encoded objects. *CG&A*, 1:3–29, 1981.

SP Dunstan y AJB Mill. Spatial indexing of geological models using linear octrees. *Computers & Geosciences*, 15(8):1291–1301, 1989.

Dun Fletcher y Ian Parberry. *3D Math Primer for Graphics and Game Development*. Word Ware Publishing, Inc., 2002.

Sarah F Frisken y Ronald N Perry. Simple and efficient traversal methods for quadtrees and octrees. *Journal of Graphics Tools*, 7(3):1–11, 2002.

Ronald L Graham. *Concrete mathematics:[a foundation for computer science; dedicated to Leonhard Euler (1707-1783)]*. Pearson Education India, 1994.

Charles AR Hoare. Quicksort. *The Computer Journal*, 5(1):10–16, 1962.

Christoph M Hoffmann. *Geometric and solid modeling*. Morgan Kaufmann, 1989.

Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, y Werner Stuetzle. Mesh optimization. En *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, págs. 19–26. ACM, 1993.

David V Hutton y Jianhua Wu. *Fundamentals of finite element analysis*, tomo 1. McGraw-Hill New York, 2004.

Djaffer Ibaroudene y Raj Acharya. Parallel display of objects represented by linear octrees. *Parallel and Distributed Systems, IEEE Transactions on*, 6(1):79–85, 1995.

C. L. Jackins y S. L. Tanimoto. Octrees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, 14:249–270, 1980.

Barry Joe. Quadrilateral mesh generation in polygonal regions. *Computer-Aided Design*, 27(3):209–222, 1995.

Nicolai M Josuttis. *The C++ standard library: a tutorial and reference*. Addison-Wesley, 2012.

Donald Ervin Knuth. *The art of computer programming: sorting and searching*, tomo 3. Pearson Education, 1998.

Arístides Legrá. *La calidad de la información en el proceso de la automatización de la minería del niquel en Cuba*. Moa : Minería y Geología, 2003.

Thomas Lewiner, Vinícius Mello, Adelailson Peixoto, Sinésio Pesco, y Hélio Lopes. Fast generation of pointerless octree duals. En *Computer Graphics Forum*, tomo 29, págs. 1661–1669. Wiley Online Library, 2010.

Donald Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147, 1982.

Dinesh P Mehta y Sartaj Sahni. *Handbook of data structures and applications*. CRC Press, 2004.

Matthew Moore y Jane Wilhelms. Collision detection and response for computer animation. *ACM Siggraph Computer Graphics*, 22(4):289–298, 1988.

Ananth Potty. Efficient ray casting. *Masters thesis, Indian Institute of Technology*, 1997.

Carlos Quintana. Algoritmo para la creación de volúmenes de restricción para búsqueda mínima en modelos de bloques geológicos. *Universidad de las Ciencias Informáticas*, 2013.

Aristides AG Requicha y Herbert B Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE*, 73(1):30–44, 1985.

Héctor Rodríguez. El banco de datos geológicos de la República de Cuba. Bases conceptuales y tecnológicas para su informatización. En *Quinto Congreso de Geofísica. La Habana, Cuba*. 2009.

David F Rogers. *An introduction to NURBS: with historical perspective*. Elsevier, 2000.

D. H. Ruiz. El tratamiento automatizado de la información geológica en los yacimientos de níquel cubanos. En *Quinto Congreso de Geofísica. La Habana, Cuba*. 2009.

Hanan Samet. *The design and analysis of spatial data structures*, tomo 85. Addison-Wesley Reading, MA, 1990.

Michael Ian Shamos y Dan Hoey. Geometric intersection problems. En *Foundations of Computer Science, 1976., 17th Annual Symposium on*, págs. 208–215. IEEE, 1976.

Raj Shekhar, Elias Fayyad, Roni Yagel, y J Fredrick Cornhill. Octree-based decimation of marching cubes surfaces. En *Visualization'96. Proceedings.*, págs. 335–342. IEEE, 1996.

Hari Sundar, Rahul S Sampath, y George Biros. Bottom-up construction and 2: 1 balance refinement of linear octrees in parallel. *SIAM Journal on Scientific Computing*, 30(5):2675–2708, 2008.

Tiankai Tu, David R O'Hallaron, y Julio Lopez. The etree library: A system for manipulating large octrees on disk. 2003.

Yusnier Valle. Modelación y visualización de superficies de terrenos en tres dimensiones. *Universidad de las Ciencias Informáticas*, 2009.

John Vince. *Geometric algebra for computer graphics*. Springer Science & Business Media, 2008.

Roger Weber, Hans-Jörg Schek, y Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. En *VLDB*, tomo 98, págs. 194–205. 1998.

Rui Xu y Don Wunsch. *Clustering*, tomo 10. John Wiley & Sons, 2008.

K Yamaguchi, TL Kunii, K Fujimura, y H Toriya. Octree-related data structures and algorithms. *IEEE Computer Graphics and Applications*, 4(1):53–59, 1984a.

K Yamaguchi, TL Kunii, David F Rogers, Steven G Satterfield, y Francisco A Rodriguez. Computer-integrated manufacturing of surfaces using octree encoding. *IEEE Computer Graphics Applications*, 4:60–62, 1984b.

S Zapotocky. Image-based modeling with polyhedral primitives. 2013.